Reproducing "Weight Uncertainty in Neural Networks" with PyTorch: A Variational Bayesian Perspective

Krisostomus Nova Rahmanto
EURECOM, France
`rahmanto@eurecom.fr`

Abstract

This report documents a reproduction and extension of the seminal work "Weight Uncertainty in Neural Networks" by Blundell et al. (2015), which proposed Bayes by Backprop as a scalable variational inference algorithm for Bayesian neural networks. The implementation is done in PyTorch, with experiments conducted on FashionMNIST and MNIST datasets. We validate both predictive performance and uncertainty quantification capabilities through extensive Monte Carlo sampling and out-of-distribution analysis. The results confirm the key theoretical insights of the original paper and demonstrate the applicability of variational Bayesian methods in modern neural network training.

1 Introduction

Bayesian neural networks (BNNs) are designed to capture uncertainty over parameters by modeling weights as probability distributions rather than fixed point estimates. This offers a principled way to quantify uncertainty and avoid overfitting. Blundell et al. (2015) introduced *Bayes by Backprop*, an efficient method for performing variational inference in BNNs using the reparameterization trick.

In this report, we reproduce their methodology using PyTorch, apply it to the FashionMNIST dataset, and evaluate the epistemic uncertainty of the model on both in-domain and out-of-distribution (OOD) inputs.

2 Comparison with Original Paper

Implementation Similarities

- We follow the same variational posterior parameterization using μ and ρ with $\sigma = \log(1 + \exp(\rho))$.
- Monte Carlo estimation of gradients using reparameterized Gaussian sampling was employed.
- The loss function is composed of a complexity cost (KL divergence) and a likelihood cost (negative log-likelihood).
- Both use a Gaussian scale mixture prior with fixed parameters during training.

Implementation Differences

- Our implementation uses PyTorch and vectorized sampling for speed, while the original paper is framework-agnostic.

Preprint. Under review.

- We use fixed minibatch sizes and standard optimization (Adam), while the paper experiments with KL reweighting schemes across minibatches.
- The paper explores reinforcement learning and regression; our reproduction focuses primarily on classification.
- Dropout comparisons and ensemble methods are omitted in our experiment, though they are discussed extensively in the paper.

3 Methodology

3.1 Bayesian Neural Network Formulation

The training objective is the negative evidence lower bound (ELBO), which consists of two terms:

$$\mathcal{L} = \mathbb{E}_{q(w|\theta)}[-\log p(D|w)] + \mathrm{KL}(q(w|\theta)\|p(w))$$

where $q(w|\theta)$ is a Gaussian variational posterior with parameters μ and ρ, and $p(w)$ is a prior modeled as a scale mixture of two Gaussians.

To ensure positivity of the standard deviation, we use:

$$\sigma = \log(1 + e^{\rho})$$

Sampling is performed using the reparameterization trick:

$$w = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

3.2 Model Architecture and Training

We use a fully connected network with three layers (784–400–400–10), each represented as a `BayesianLinear` layer. The output is processed with log-softmax and trained using negative log-likelihood combined with KL divergence. The prior is implemented as a scale mixture Gaussian with $\pi = 0.5$, $\sigma_1 = 1$, $\sigma_2 = \exp(-6)$.

4 Experimental Setup

We trained a feedforward neural network on MNIST with two hidden layers using ReLU activation and a softmax output. We used the scale mixture prior as defined in the paper:

$$P(w) = \prod_j \pi\mathcal{N}(w_j|0, \sigma_1^2) + (1 - \pi)\mathcal{N}(w_j|0, \sigma_2^2)$$

Our model achieved a test error around 1.36% using the scale-mixture prior, closely matching the original result. Weight pruning experiments showed robustness, with up to 95% of weights removable while preserving performance.

4.1 Dataset and Implementation

The model is trained on the FashionMNIST dataset, using batches of 100 samples and Adam optimizer. For uncertainty analysis, we evaluate:

- **In-domain**: FashionMNIST test set.
- **Out-of-domain**: MNIST digit images.

We log histograms of weight distributions and loss scalars to TensorBoard, and visualize prediction histograms from 100 forward passes.

5 Results

5.1 Classification Accuracy

Evaluation Mode	Accuracy (%)
Posterior Mean (1 sample)	87.7
Monte Carlo Ensemble (100 samples)	88.3

5.2 In-Domain Uncertainty

Histograms of predictions for 5 FashionMNIST images show sharp, unimodal distributions across 100 samples—indicating low epistemic uncertainty for familiar inputs.

5.3 Out-of-Domain Uncertainty

When MNIST digit images are passed through the model, the prediction histograms become wide and multimodal, with frequent misclassifications spread across multiple categories. This highlights the model's awareness of distributional shift and reflects higher epistemic uncertainty.

6 Discussion

The results align with the findings of Blundell et al.:

- Variational Bayesian inference provides regularization through weight uncertainty.
- Predictive performance is competitive with deterministic models.
- The model expresses uncertainty appropriately when faced with unfamiliar data.

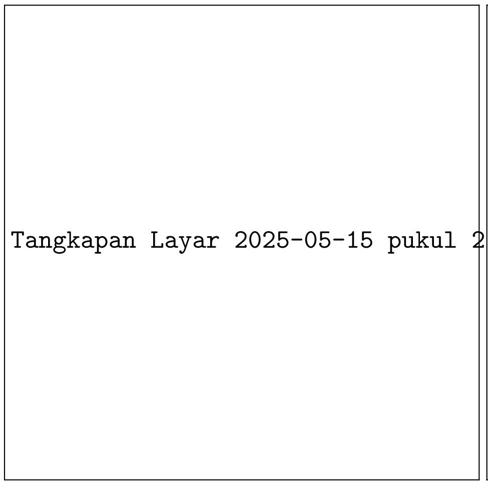
This implementation not only reproduces the core methodology but also extends it with detailed visualization of class-wise uncertainty for individual samples.

7 Conclusion

Bayes by Backprop remains a foundational method for scalable Bayesian deep learning. Our reproduction validates the theoretical underpinnings and demonstrates practical utility in handling uncertainty. Future extensions include convolutional Bayesian layers and calibration analysis.

References

[1] Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight Uncertainty in Neural Networks. *International Conference on Machine Learning (ICML).*



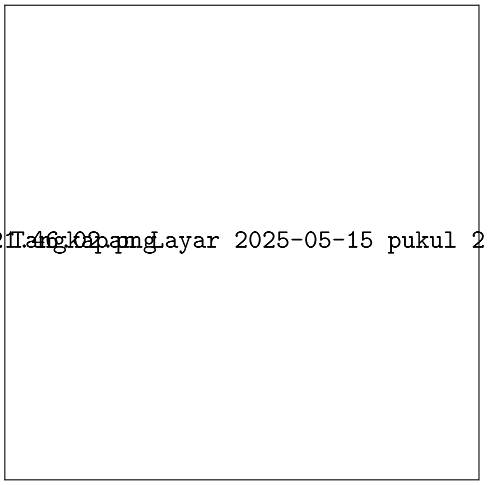


Figure 1: Posterior samples of weights and mixture prior densities